In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from sklearn.model_selection import StratifiedKFold
import xgboost as xgb

import tensorflow as tf

pd.options.display.max_columns = None
```

In [ ]:
```
df_covid = pd.read_csv('./Covid_clean.csv')
df_covid.head()
```

```
C:\Users\ismael\AppData\Local\Temp\ipykernel_28656\3897149584.py:1:
DtypeWarning: Columns (4,20) have mixed types. Specify dtype option
on import or set low_memory=False.
  df_covid = pd.read_csv('./Covid_clean.csv')
```

Out[ ]:

| | USMER | MEDICAL_UNIT | SEX | PATIENT_TYPE | DATE_DIED | PNEUMONI |
|---|---|---|---|---|---|---|
| **0** | 2 | 1 | 1 | 1 | 2020-05-03 | 1 |
| **1** | 2 | 1 | 0 | 1 | 2020-06-03 | 1 |
| **2** | 2 | 1 | 0 | 0 | 2020-06-09 | 0 |
| **3** | 2 | 1 | 1 | 1 | 2020-06-12 | 0 |
| **4** | 2 | 1 | 0 | 1 | 2020-06-21 | 0 |

In [ ]:
```
df_covid.shape
```

Out[ ]:
```
(1024829, 21)
```

In [ ]:
```
# creamos el modelo de clasificacion

features = ['USMER', 'SEX', 'PNEUMONIA', 'AGE', 'DIABETES', 'COPD'
target = 'fallecidos'
```

In [ ]:
```
df_covid['fallecidos'].unique()
```

Out[ ]:
```
array([1, 0], dtype=int64)
```

# Rebalanceo y xGboost

In [ ]:
```
X = df_covid[features]
y = df_covid[target]
```

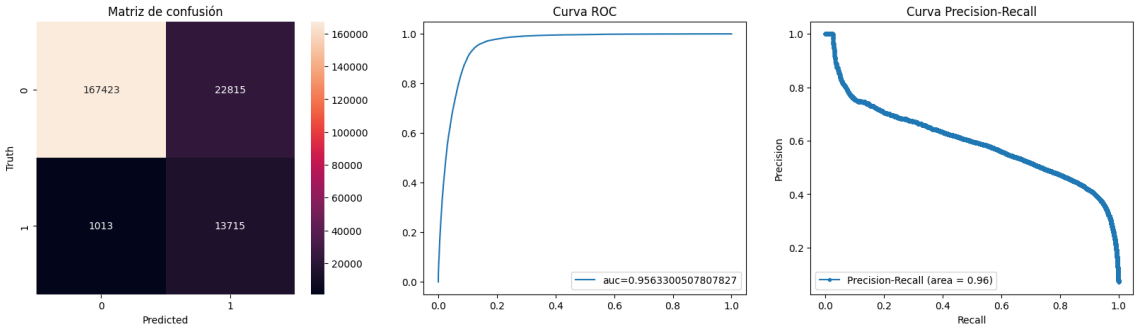
In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz



model = xgb.XGBClassifier(scale_pos_weight=len(y_train[y_train==0]
model.fit(X_train, y_train)

# Definición de los hiperparámetros a ajustar
param_grid = {
    'learning_rate': [0.1, 0.01, 0.001],
    'max_depth': [3, 5, 7],
    'n_estimators': [100, 200, 300],
}

# Instancia de Grid Search Cross Validation
grid_search = GridSearchCV(model, param_grid, scoring='accuracy',

# Entrenamiento del modelo con Grid Search
grid_search.fit(X_train, y_train)

# Mejores hiperparámetros encontrados
best_params = grid_search.best_params_
print("Mejores hiperparámetros:", best_params)

# Evaluación del modelo con los mejores hiperparámetros en el conju
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

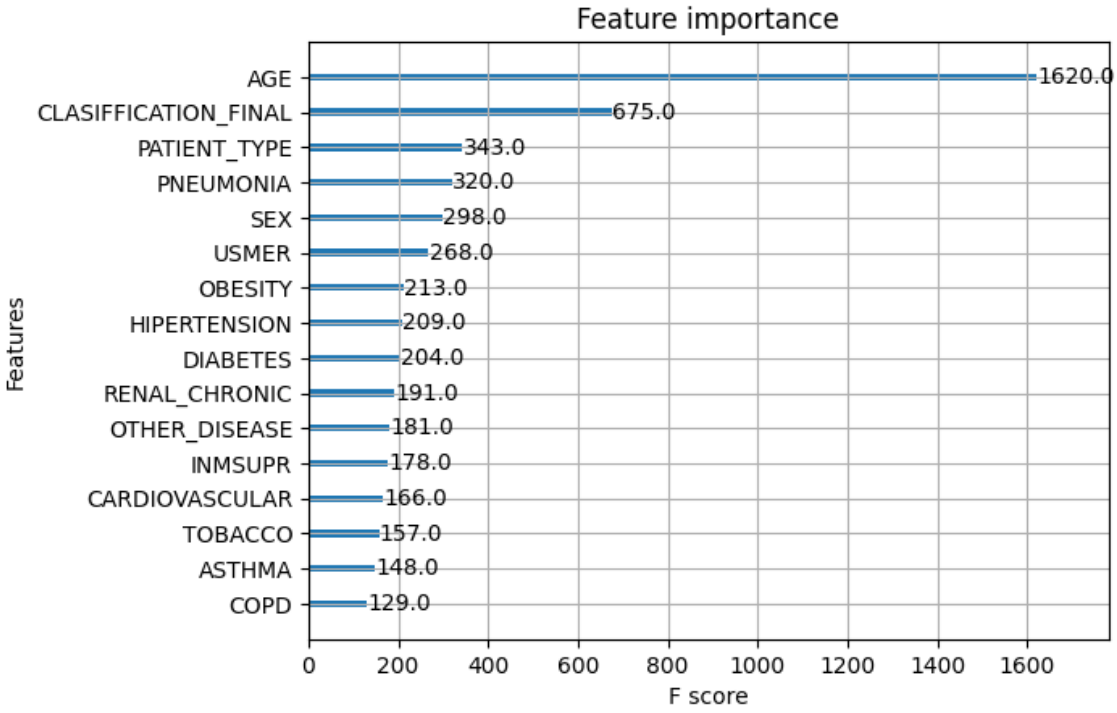
```
Mejores hiperparámetros: {'learning_rate': 0.001, 'max_depth': 3,
'n_estimators': 200}
Accuracy: 0.8837465726022853
```

In [ ]:
```
print(classification_report(y_test, y_pred))

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

precision = precision_score(y_test, y_pred)
print("Precision:", precision)
```

```
recall = recall_score(y_test, y_pred)
print("Recall:", recall)

f1 = f1_score(y_test, y_pred)
print("F1:", f1)


# grafiocamos la matriz de confusión

cm = confusion_matrix(y_test, y_pred)

print(f'Matriz de confusión: {cm}')
# Curva ROC

y_pred_proba = model.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test,  y_pred_proba)
auc = roc_auc_score(y_test, y_pred_proba)

# Curva Precision-Recall

precision, recall, _ = precision_recall_curve(y_test, y_pred_proba

# ploteo los 3 graficos en un mosaico

fig, ax = plt.subplots(1, 3, figsize=(20, 5))

sns.heatmap(cm, annot=True, fmt='d', ax=ax[0])
ax[0].set_xlabel('Predicted')
ax[0].set_ylabel('Truth')
ax[0].set_title('Matriz de confusión')

ax[1].plot(fpr,tpr,label="auc="+str(auc))
ax[1].legend(loc=4)
ax[1].set_title('Curva ROC')

ax[2].plot(recall, precision, marker='.', label='Precision-Recall
ax[2].legend(loc="lower left")
ax[2].set_xlabel('Recall')
ax[2].set_ylabel('Precision')
ax[2].set_title('Curva Precision-Recall')

xgb.plot_importance(model)

plt.show()


plt.show()
```

```
                precision    recall  f1-score   support

           0       0.99      0.88      0.93    190238
           1       0.38      0.93      0.54     14728

    accuracy                           0.88    204966
   macro avg       0.68      0.91      0.73    204966
weighted avg       0.95      0.88      0.90    204966

Accuracy: 0.8837465726022853
Precision: 0.37544483985765126
Recall: 0.9312194459532862
F1: 0.5351359787740451
Matriz de confusión: [[167423  22815]
 [  1013  13715]]
```




# Analisis de falsos negativos

In [ ]:
```
# buscamos los falsos negativos y armamos un df con ellos

falsos_negativos = np.where((y_test == 1) & (y_pred == 0))[0]
```

```
# Busco los indices de los falsos negativos
indices = X_test.iloc[falsos_negativos].index

# Armo un df con los falsos negativos

df_falsos_negativos = df_covid.loc[indices]

df_falsos_negativos['falsos_negativos'] = y_pred[falsos_negativos]

features = ['USMER', 'SEX', 'PNEUMONIA', 'AGE', 'DIABETES', 'COPD'

df_falsos_negativos[features].head(10)
```

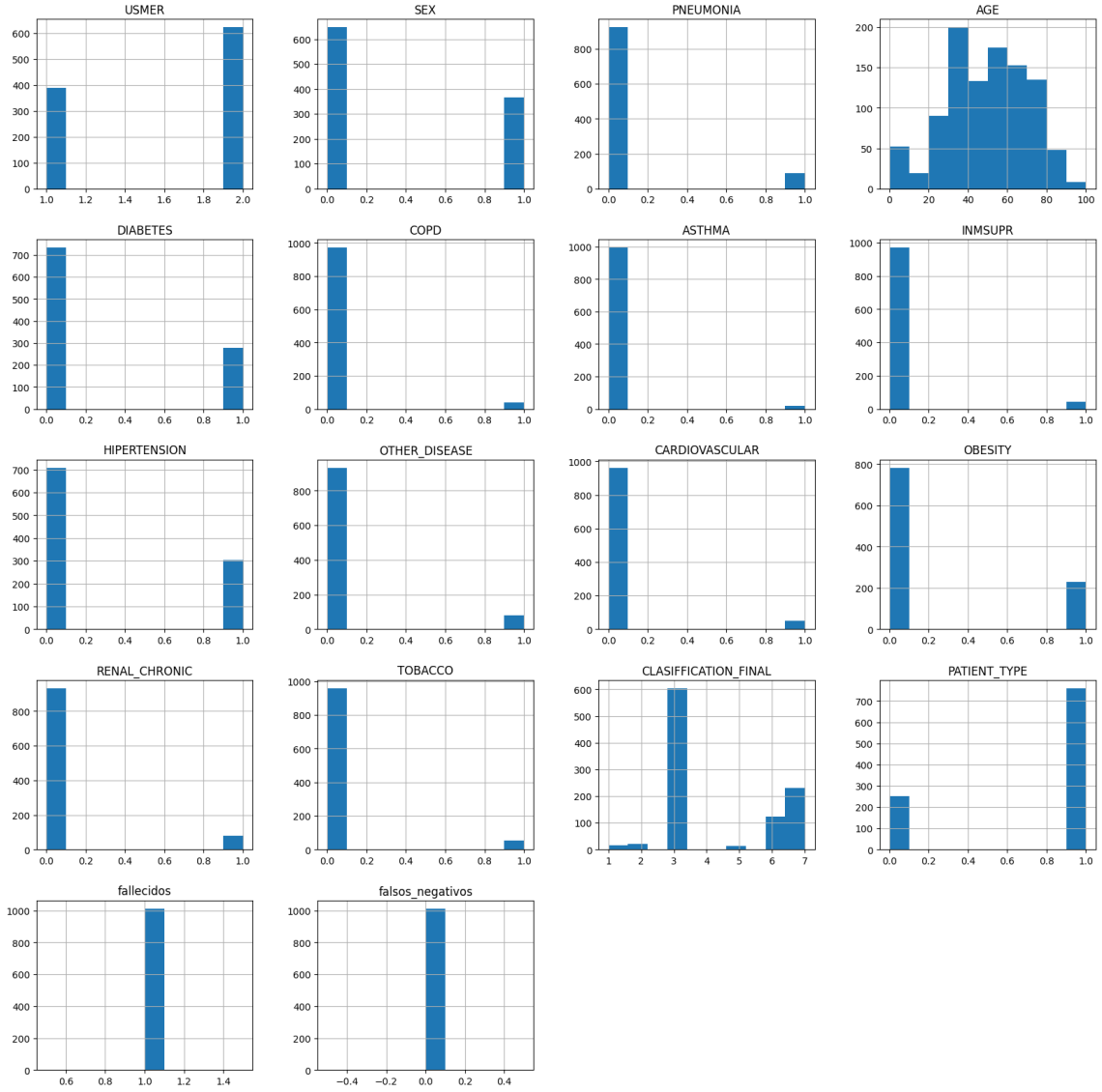
Out[ ]:

| | USMER | SEX | PNEUMONIA | AGE | DIABETES | COPD | ASTHMA |
|---|---|---|---|---|---|---|---|
| **36804** | 1 | 0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| **48941** | 1 | 1 | 0.0 | 50.0 | 0.0 | 1.0 | 0.0 |
| **334520** | 2 | 0 | 0.0 | 25.0 | 0.0 | 0.0 | 0.0 |
| **49932** | 2 | 0 | 0.0 | 36.0 | 0.0 | 0.0 | 0.0 |
| **448225** | 2 | 0 | 1.0 | 25.0 | 0.0 | 0.0 | 0.0 |
| **52971** | 1 | 1 | 0.0 | 77.0 | 1.0 | 1.0 | 0.0 |
| **53340** | 1 | 0 | 0.0 | 77.0 | 0.0 | 0.0 | 0.0 |
| **44461** | 1 | 0 | 0.0 | 64.0 | 1.0 | 0.0 | 0.0 |
| **451354** | 2 | 1 | 0.0 | 69.0 | 1.0 | 0.0 | 0.0 |
| **441326** | 2 | 1 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |

In [ ]:

```
# graficamos las variables de df_falsos_negativos

df_falsos_negativos[features].hist(figsize=(20,20))
# agrego titulo

plt.suptitle('Histogramas de variables de los falsos negativos', f
plt.savefig('histogramas_falsos_negativos.png')
plt.show()
```

Histogramas de variables de los falsos negativos




In [ ]:
```
# buscamos los falsos positivos y armamos un df con ellos

falsos_positivos = np.where((y_test == 0) & (y_pred == 1))[0]

# Busco los indices de los falsos negativos
indices = X_test.iloc[falsos_positivos].index

# Armo un df con los falsos negativos

df_falsos_positivos = df_covid.loc[indices]

df_falsos_positivos['falsos_negativos'] = y_pred[falsos_positivos]

features = ['USMER', 'SEX', 'PNEUMONIA', 'AGE', 'DIABETES', 'COPD'

df_falsos_positivos[features].head(10)
```

Out[ ]:

| | USMER | SEX | PNEUMONIA | AGE | DIABETES | COPD | ASTHMA |
|---|---|---|---|---|---|---|---|
| **271938** | 2 | 1 | 1.0 | 49.0 | 0.0 | 0.0 | 0.0 |
| **10968** | 1 | 1 | 1.0 | 49.0 | 1.0 | 0.0 | 0.0 |
| **217922** | 1 | 1 | 1.0 | 47.0 | 0.0 | 0.0 | 0.0 |
| **375447** | 1 | 0 | 1.0 | 46.0 | 0.0 | 0.0 | 0.0 |
| **608948** | 1 | 1 | 0.0 | 65.0 | 1.0 | 0.0 | 0.0 |
| **4416** | 2 | 0 | 1.0 | 43.0 | 0.0 | 0.0 | 0.0 |
| **239681** | 2 | 1 | 1.0 | 47.0 | 1.0 | 0.0 | 0.0 |
| **328379** | 1 | 0 | 1.0 | 78.0 | 0.0 | 0.0 | 0.0 |
| **846235** | 1 | 1 | 1.0 | 4.0 | 0.0 | 0.0 | 0.0 |
| **344222** | 2 | 0 | 0.0 | 45.0 | 1.0 | 0.0 | 0.0 |

In [ ]:

```
df_falsos_positivos[features].hist(figsize=(20,20))
# agrego titulo

plt.suptitle('Histogramas de variables de los falsos positivos', f
plt.savefig('histogramas_falsos_positivos.png')
plt.show()
```

Histogramas de variables de los falsos positivos